BigSUR: Large-scale Structured Urban Reconstruction

TOM KELLY, University College London
JOHN FEMIANI, Miami University
PETER WONKA, KAUST
NILOY J. MITRA, University College London

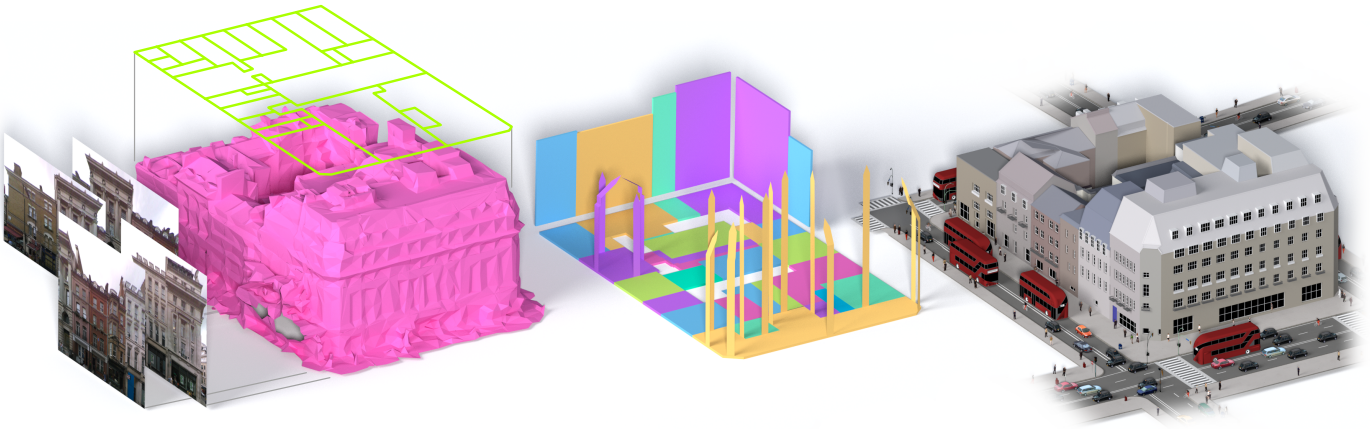


Fig. 1. *Structured Urban Reconstruction.* Given street-level imagery, GIS footprints, and a coarse 3D mesh (left), we formulate a global optimization to automatically fuse these noisy, incomplete, and conflicting data sources to create building footprints (middle: colored horizontal polygons) with profiles (vertical ribbons shown for several footprints) and attached building façades (vertical rectangles). The output encodes a structured urban model (right) including the walls, roof, and associated building elements (e.g., windows, balconies, roof, wall color, etc.). Inset below: A reference aerial image.

The creation of high-quality semantically parsed 3D models for dense metropolitan areas is a fundamental urban modeling problem. Although recent advances in acquisition techniques and processing algorithms have resulted in large-scale imagery or 3D polygonal reconstructions, such data-sources are typically noisy, and incomplete, with no semantic structure. In this paper, we present an automatic data fusion technique that produces high-quality structured models of city blocks. From coarse polygonal meshes, street-level imagery, and GIS footprints, we formulate a binary integer program that globally balances sources of error to produce semantically parsed mass models with associated façade elements. We demonstrate our system on four city regions of varying complexity; our examples typically contain densely built urban blocks spanning hundreds of buildings. In our largest example, we produce a structured model of 37 city blocks spanning a total of 1,011 buildings at a scale and quality previously impossible to achieve automatically.

CCS Concepts: • **Computing methodologies → Scene understanding**; *Shape analysis*; *Mesh models*; • **Applied computing → Architecture (buildings)**;

Additional Key Words and Phrases: urban modeling, structure, reconstruction, façade parsing and element classification, procedural modeling

0730-0301/2017/11-ART204 $15.00
https://doi.org/10.1145/3130800.3130823

ACM Reference format:
Tom Kelly, John Femiani, Peter Wonka, and Niloy J. Mitra. 2017. BigSUR: Large-scale Structured Urban Reconstruction. *ACM Trans. Graph.* 36, 6, Article 204 (November 2017), 16 pages.
https://doi.org/10.1145/3130800.3130823

1 INTRODUCTION

Obtaining detailed 3D urban models is important for a variety of applications ranging from urban planning and environmental simulations to virtual reality and video game creation. Given the importance of such models, extensive efforts have been undertaken to create polygonal meshes from aerial images or light detection and ranging (LiDAR) scans. Such datasets are often very expensive and tedious to create. They are difficult to use because they are typically heterogeneous with sparse or missing details. More importantly, they lack semantic structure, which prevents easy use in subsequent applications.

In contrast, procedural pipelines (e.g., CityEngine) create homogeneous, semantically labelled urban models. One such procedural pipeline uses horizontal (building) footprints and the corresponding

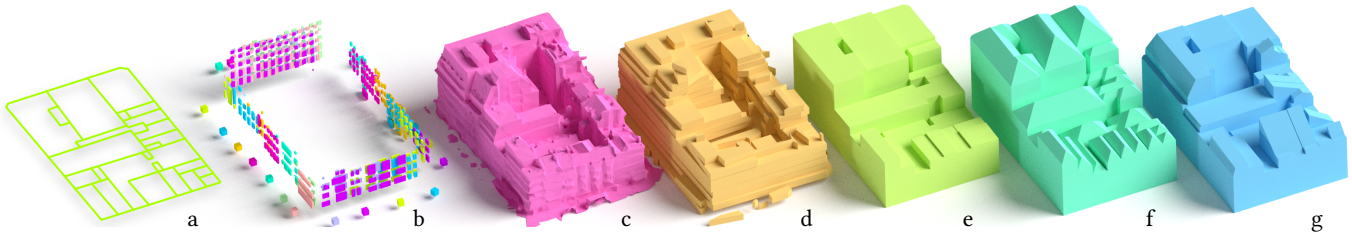


Fig. 2. *Baseline methods.* (a) GIS footprints represent plot ownership more accurately than building structure. (b) Image features, such as windows (cuboids) extracted from street-level imagery, are available only near where the images have been taken (cubes), and lack information about the interior of the structure; different images may give contradictory features for the same building. (c) Raw polygonal meshes tend to be more complete, but they contain noise and are typically polygon soups. One reconstruction possibility is to fit horizontal "floors" to the mesh (d), while another is to extrude the GIS footprints to heights available from a database (e). Both these approaches fail to convey the roof structures of the input. A popular GIS data visualization techniques is to create a *hip roof* over all footprints (f), which leads to a monotonous structure. (g) Naively applying profiles from the input mesh to the GIS footprints leads to more interesting roof shapes; but these are inaccurate because the GIS edges are frequently not representative of real-world building walls.

vertical profiles to create mass models by extruding the footprint upwards along the profiles, which may then be 'decorated' with building elements such as windows, doors, etc. Currently, this workflow is suitable for coarse approximation of larger areas, or for detailed manual modeling of particular (iconic) buildings, but it does not scale to accurate detailed modeling of wider urban areas.

In this paper, we focus on the problem of procedurally creating structured models by leveraging data from multiple sources (see Figure 1 and the inset aerial view for reference). Such raw information has different strengths and weaknesses: for example, publicly available *Geographic Information System footprints* (GIS footprints) carry reliable records of plot ownership, but they often do not reflect built reality; polygonal meshes, often in the form of polygon soups obtained by processing aerial images, provide coarse information, but they lack semantic partitioning or fine details; street-level imagery (e.g., façade photographs) provides detailed information, but it lacks 3D information or semantic labels. Further, each data source has its own coordinate system, suffers from distortion, and frequently contains mutually conflicting or partial information.

Naively combining information across the above datasources results in various types of artifacts (see Figure 2). For example, extruding GIS footprints with profiles extracted from mesh data creates misleading mass models, while transferring window locations regressed from images onto estimated façade planes results in poorly positioned windows.

Instead of heuristically combining the above datasources, we propose a unified fusion algorithm. We develop an optimization formulation that analyzes the heterogeneous data sources (i.e., GIS footprints, polygonal meshes, and street-level imagery) and retargets them to a single consistent representation. By balancing the various retargeting costs, our algorithm reaches a consensual *structured model*, the output of which is building-level footprints, associated profiles along the footprint boundaries, and façade elements placed appropriately over the mass models (see Figure 1). The raw input data to our algorithm comes from various preferred layout directions (extracted from GIS information), candidate building footprints and profiles (extracted from the polygonal meshes), and façade partitions with associated elements (extracted by analyzing the individual façade images). Our system automatically decides *which* of these elements to retain and *how* to adapt the selected elements to create consistent output. Figure 15 shows the input GIS footprints and the extracted building footprints produced by our algorithm. We note that the result is semantically structured in the sense that the output has labels associated with the different sections of the output model (e.g., windows, balconies, shops, walls, roofs, etc.). Further, our algorithm does *not* make Manhattan-world assumptions, nor does it restrict the roof angles (i.e., roofs can be flat or sloped), nor number of pitches (i.e., façades can alternate an arbitrary number of times between wall and roof).

We demonstrate the effectiveness of our system by evaluating four differing urban settings: *Detroit* as a suburban US city with simple detached houses, *New York* with blocks of near-regular high-rise buildings arranged on a (literal) Manhattan-grid, *Oviedo* as a typical historic European city with non-axis aligned buildings surrounding inner courtyards, and *London* with dense urban architecture with many annexes and complex roof shapes. Finally, we semantically reconstruct a very large area of central London covering 37 blocks around Oxford Circus and compare our method with state-of-the-art urban reconstruction techniques.

In summary, we introduce a novel wide-area fusion algorithm that semantically combines multi-channel, noisy, and conflicting information to produce structured models in the form of building mass models with associated façade elements. We demonstrate the automated method on urban neighborhoods spanning several building blocks at a scale that has not been previously demonstrated.

2 RELATED WORK

We review the relevant literature on the urban modeling and reconstruction pipeline (see [Musialski et al. 2013] for a survey).

2.1 Reconstructing mass models

There are multiple possible inputs for large-scale urban mass modeling. Mass models are often reconstructed from aerial images or LiDAR [Brenner 2005]. Other modalities, such as synthetic aperture radar (SAR), ground based photographs, or videos, are less common. Furthermore, satellite data have lower resolution and drones can capture only smaller areas. While LiDAR produces point clouds directly, images must be processed to produce sparse [Snavely et al.

2006] or dense [Ceylan et al. 2013; Furukawa and Ponce 2010] point clouds. Some integrated modeling pipelines extract mass models from images directly [Dick et al. 2004; Garcia-Dorado et al. 2013; Vanegas et al. 2010]. Surface models can be extracted from point clouds, e.g., by resampling onto a grid [Poullis and You 2009], 2.5D contouring [Zhou and Neumann 2010], relation-based primitive fitting [Monszpart et al. 2015], or Poisson reconstruction [Kazhdan and Hoppe 2013]. Another important component in urban modeling is segmentation [e.g., Golovinskiy et al. 2009; Matei et al. 2008; Verdie et al. 2015] to separate buildings from other classes.

Our work is mainly related to shape abstraction and simplification; we aim to create simple and plausible mass models from noisy input data. One simple model for shape abstraction is to regularize the models using the Manhattan-world assumption [Li et al. 2016]. Alternately, very good results can be achieved by fitting parametric building blocks to height fields [Lafarge et al. 2010] or LiDAR input [Lin et al. 2013], exploiting non-local regularity relations [Zheng et al. 2010], or obtaining depth-layer relations by jointly analyzing images and LiDAR scans [Li et al. 2011b]. Following Verdie et at. [2015], we use a noisy building mesh as input. They use a simplified version of Globfit [Li et al. 2011a] to detect relationships between extracted planes to regularize the output. In contrast to this method, we jointly analyze the different input data modalities to produce a consistent structured model, in which, for example, the footprints of the mass models are in agreement with how the street-level imagery is partitioned into different buildings.

2.2 Façade parsing

The goal of façade parsing is to extract façade elements such as windows, doors, and balconies. The input of façade parsing is typically a single image or a point cloud. A typical initial step of façade parsing is to compute local per-pixel information, such as segmentation information [Martinović et al. 2012], edge detection, or symmetry detection [Müller et al. 2007]. This input is then regularized to make it more compliant with a given model of a façade structure [Cohen et al. 2014]. One possible model is a grid with one spacing parameter for each row and each column [Müller et al. 2007], which can also be represented by a rank-one matrix [Yang et al. 2012]. A more general model is a hierarchical splitting tree, in which each internal node splits into multiple horizontal or vertical slices [Dai et al. 2012; Kozinski et al. 2015; Riemenschneider et al. 2012; Shen et al. 2011; Teboul et al. 2013]. These hierarchical approaches differ in how they incorporate low-level features stemming from classifiers and in how they use encoded architectural knowledge. Example solutions include use of MRFs [Kozinski et al. 2015], extending the CYK algorithm [Riemenschneider et al. 2012], application of reinforcement learning [Teboul et al. 2013], post-processing by optimization [Jiang et al. 2016; Martinović et al. 2012; Nan et al. 2015], or jointly optimizing for template matching and deformation estimation [Ceylan et al. 2016]. A significant simplification used by these systems is to consider only façade images that have been rectified and cropped for individual buildings.

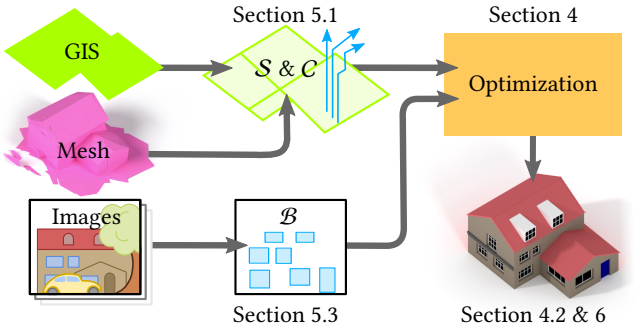


Fig. 3. *Overview.* Starting from GIS footprints, a coarse 3D mesh, and street-level imagery, we extract a set of sweep-edges, $\mathcal{S}$, a set of clean-profiles, $\mathcal{C}$, and a set of building-façades, $\mathcal{B}$. These are then globally optimized to produce a semantically parsed building block as output.

2.3 Interactive reconstruction

To achieve improved results, another line of work investigates interactive techniques for mass modeling [Debevec et al. 1996], or façade parsing. For example, Nan et al. present an interactive façade modeling system for LiDAR data [2010] and Xiao et al. propose an interactive system for images [2008]. Another recent concept is to train multiple neural networks to interactively create procedural models from input sketches [Nishida et al. 2016]. In contrast, we aim to create an automatic system.

In this work, we build on the geometry of the straight skeleton [Aichholzer et al. 1996] to model architecture. Early work used the *unweighted* straight skeleton to model roofs [Laycock and Day 2003; Müller et al. 2006] and walls [Fang et al. 2013]. The *weighted* skeleton [Eppstein and Erickson 1999] offered enhanced expressiveness; in particular, the *procedural extrusion* system (PE) [Kelly and Wonka 2011] consisted of stacked weighted skeletons. Recently, Biedl et al. [2016] reinforced the theoretical underpinnings of the weighted straight skeleton, renewing our interest in PEs. Essentially, PEs are a parameterization of architecture into a horizontal 2D plan with a set of vertical 2D profiles that are associated with the edges of this plan. Such a parameterization can represent buildings with arbitrarily angled walls and roofs to provide a strong architectural prior. In this work, we develop a method to project real-world data into the space of buildings represented by PEs.

3 PROBLEM SETUP

Our system takes input from three sources — publicly available GIS footprints, a coarse 3D mesh, and street-level façade images — with the goal of reconstructing a high-quality semantic model of an urban area. Since the different input sources have complementary strengths and weaknesses, we first process them individually to extract three types of entities: *sweep-edges*, *clean-profiles*, and *building-façades*. In the following, we describe these entities, while deferring the details of how they are computed to Section 5; the global optimization, which fuses them to produce the structured model, is discussed in Section 4. Figure 3 presents an overview of our framework.

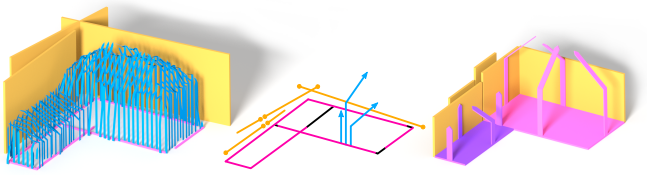


Fig. 4. *Terminology.* Left: The data are used to create street-level imagery with associated façade planes (orange), raw-profiles (blue) and sweep-edges (pink). Center: These are processed to create the input to the optimization — a smaller set of clean-profiles (blue), building-façades (orange lines), and building-façade-points (orange points), and a ground plane tessellation consisting of sweep-edges (pink) and soft-edges (black) enclosing faces. Right: The output of the optimization is a collection of watertight footprint-polygons (pink and purple), with a clean-profile assigned to every edge, and positions for every building-façade (orange).

3.1 GIS footprints

Typically, an urban building block consists of several densely packed buildings (up to 100 buildings in our examples). While GIS footprints (see [Miller et al. 2017]) provide an accurate ownership record, surprisingly they provide little usable information concerning a building's physical walls and partitions, making it challenging to use these data directly for reconstruction. However, we found that they carry a mixture of accurate and noisy orientation information, which we utilize to regularize the processing of other data sources.

3.2 Coarse 3D mesh

A 3D mesh or polygon soup (e.g., obtained via multi-view stereo or LiDAR scans) provides approximate, incomplete, noisy, but large-scale geometric information. We process such meshes to produce two entities: horizontal *sweep-edges* and vertical *clean-profiles* (see Figure 4); such sweep-edges are extruded along clean-profiles to create a mass model. Specifically, we extract a set of lines, referred to as sweep-edges, $\mathcal{S}$, on the ground-plane by identifying likely façades over the mesh. Along these sweep-edges, we vertically slice the mesh to create many *raw-profiles*; these are clustered, averaged, and abstracted to create a *set* of *clean-profiles*, C (see Figure 4 and Section 5.1). Direct reconstruction from these sweep-edges and clean-profiles is challenging as PEs require watertight footprint-polygons, with a clean-profile assigned to each edge. Specifically, there are two sources of difficulty: the sweep-edges have gaps, may self-intersect, or even be missing entirely in regions, while the clean-profiles are the output of local analysis, thus lacking information about building partitions and containing different sources of noise (e.g., from initial reconstruction, trees, or vehicles).

3.3 Street-level façade images

Complementary to the above data sources, street-level imagery provides information over portions of the urban blocks. Such images typically come with estimates of camera position and orientation. For each image, we use a convolutional neural network (CNN) based supervised classifier (see Section 5.3) to detect the rectangular bounds of a façade as well as elements such as windows, doors, and balconies. We refer to this rectangular façade containing a collection of extracted elements as a *building-façade* (see Figure 4). Each side of a city block will typically consist of multiple overlapping building-façades: one from each of the images. However, such raw building-façades, $\mathcal{B}$, may contain position and orientation errors, have inconsistent scales, sometimes overlap, or be incomplete (e.g., occluded by trees, vehicles, or scaffolding). The ground plane location of the observed start or end of a building-façade in the street-level imagery is referred to as a *building-façade-point*.

These three data sources are in three different coordinate systems, and may introduce conflicting information, making their combination challenging. Further, each is subject to reprojection and inherent noise, both within and between datasets. For example, we found that the given location and orientation of building-façades varied on different sides of a building due to GPS or GIS errors. Poor correlation between the image and 3D mesh was sometimes observed because of differing scale estimates or changes in the environment (e.g., buildings had been constructed, modified, or demolished).

3.4 Notation

Before we formulate the main binary integer program (BIP) that processes these inputs, we first introduce some notation. We use sweep-edges, $\mathcal{S}$, to oversegment the ground plane ($y = 0$) to form a tessellation of faces, $\mathbb{G}$, as described in Section 4.1. Our algorithm determines whether or not each edge, $\mathbf{e}_k \in \mathbb{G}$, should be selected, thus implicitly encoding the final building footprint-polygons. We represent this selection with a binary indicator variable, s^k, such that $s^k = 1$ if the edge, $\mathbf{e}_k$, is selected and forms part of a footprint-polygon, and $s^k = 0$ otherwise. Note that in densely built urban areas, even though adjacent buildings can share a common wall, the structures often have different heights or roofs. We encode such a situation by two, possibly different, profiles associated with the two sides of each interior wall, $\mathbf{e}^k$. (For the remainder of the paper, we discuss one such profile per edge, while the other one is similarly treated.) We denote the length of any edge, $\mathbf{e}_k$, as $\|\mathbf{e}_k\|$ and the maximum mesh height above a point on the ground plane, $(x, z) \in \mathbb{R}^2$, as $h(x, z)$.

We use logic operators (such as $\wedge, \vee, \oplus, \neg$) noting that each can be expressed in BIP constraints with additional variables (detailed in Appendix A). We will not explicitly introduce such extra variables and constraints, but we use the logic operator directly.

Unlike s^k, which is an individual binary variable, we will have cause to represent categorical variables (such as color or profile choice) using *selection vectors*. Note they are also called 'one hot vectors' in the literature. We denote a selection vector of length n as $\boldsymbol{\chi} := (\chi_1, \ldots, \chi_n)$; each element (such as χ_1) is a binary variable. Selection vectors have *exactly* one element set to one, while the others are all zero. We encode this condition with the constraint $\sum_{i=1}^{n} \chi_i = 1$. We will wish to compare two selection vectors. For example, given $\boldsymbol{\chi} := (\chi_1 \ldots \chi_n)$ and $\boldsymbol{\psi} := (\psi_1 \ldots \psi_n)$, we desire an output of 0 if all elements are equal (i.e., $\chi_i = \psi_i, \ \forall i$), and 1 otherwise. To simplify notation in this situation, we write $\textsc{isDifferent}(\boldsymbol{\chi}, \boldsymbol{\psi})$ to indicate

$$\textsc{isDifferent}(\boldsymbol{\chi}, \boldsymbol{\psi}) = (\chi_1 \oplus \psi_1) \vee \cdots \vee (\chi_n \oplus \psi_n).$$

Note that the above macro describes a set of variables and constraints to be added to the BIP.

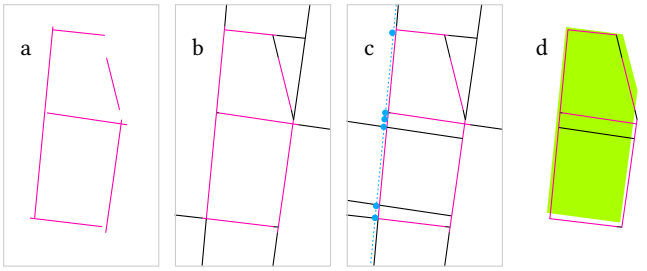


Fig. 5. *Sweep-edges and soft-edges.* A set of sweep-edges (a, pink) are extended to oversegment the ground plane (b) into faces. The sweep-edges are inserted one at a time, in order of decreasing length. To complete the tessellation, the sweep-edges are extended by *soft-edges* (black). The building-façade-points (c) further subdivide the ground plane if there are no existing similar edges. Finally, we remove faces that are mostly outside the GIS footprint (d, green) to create the tessellation, $\mathbb{G}$.

Fig. 6. *Oversegmenting the ground plane.* We use sweep-edges and GIS footprints to overpartition the ground plane. Left: The sweep-edges (pink) along with their soft-edge extensions (black) partition the plane. Center: Further oversegmentation based on the building-façades extracted from street-level imagery (blue). Right: using height and GIS information (green) we identify the interior faces to produce the oversegmentation, $\mathbb{G}$.

4 FUSION OPTIMIZATION

So far, we have introduced: (i) a set of sweep-edges, $\mathcal{S}$ (for extraction details see Section 5.1); (ii) a set of clean-profiles, C (Section 5.1); and (iii) a set of building-façades, $\mathcal{B}$ (Section 5.3). We continue to formulate a global optimization that fuses these entities to output a semantically parsed building block, simply referred to as the *structured model* (see Figure 3).

To achieve this, we address three key challenges: (i) identifying *footprint-polygons* for each building in the ground plane tessellation; (ii) selecting a clean-profile from C for each edge of every footprint-polygon; and (iii) retargeting building-façades from $\mathcal{B}$ to a subset of the edges of the footprint-polygons. A good building-façade location matches the mass models that are implicitly obtained by extruding the footprint-polygons along the selected clean-profiles.

Note that the above problems are tightly linked and must be solved together. For example, the boundary of a footprint-polygon depends on which profiles are selected, which in turn depends on how the building-façades are retargeted to match 3D mass model boundaries.

4.1 Formulation

We simultaneously address the above challenges by formulating a BIP; we next describe the optimization variables, constraints, and objective terms associated with each challenge.

4.1.1 Identifying footprint-polygons. The input GIS footprints, street-level imagery, and 3D mesh carry noisy and incomplete information about individual buildings. This is particularly pronounced in densely built urban areas where adjacent buildings often share walls, contain courtyards, and regularly break the Manhattan-world assumption. Using the available information, we first oversegment the ground plane into *faces* using the sweep-edges, then merge the oversegmented regions, and finally extract the footprint-polygons.

First, we extend the sweep-edges in $\mathcal{S}$ to initiate the ground plane oversegmentation (see Figure 5a). Note that only the edges created by sweep-edges have profiles, while others, called *soft-edges*, complete the tessellation (see Figure 5b). Next, we use the estimated building-façade-points (shown as blue dots in Figure 5c) from the street-level imagery to further oversegment the ground plane by adding soft-edges that are perpendicular to the building-façade into the tessellation. All these edges indicate potential separating walls between adjacent buildings. Finally, we discard faces that are either mostly outside the GIS footprints, or have a mean mesh height below a threshold (3m in our data). We use $\mathbb{G}$ to denote the resulting tessellation (see Figure 5d).

Extracting footprint-polygons amounts to setting the BIP variables, s^k, for each of the edges, $\mathbf{e}_k$, surrounding every face, $f_i \in \mathbb{G}$. However, setting up such an optimization is cumbersome, as not all values for $\{s^k\}$ result in valid partitions of the ground plane (see Figure 7). Hence, we indirectly formulate the problem by deciding which neighboring faces in the tessellation $\mathbb{G}$ should be merged to produce the final building footprint-polygons. For example, the resulting tessellation for Figure 1 is shown in Figure 6.

The footprint-polygons should ideally follow the sweep-edges, while making them watertight, and should use as few soft-edges as possible to fill in sections of missing data. Further, we encourage selection of edges where there is a large height difference on either side of a sweep-edge (e.g., between adjacent buildings). For each such face $f_i \in \mathbb{G}$, we sample $h(x, z)$ using the mesh data to find the mean height over the face, $h(f_i)$. This averaging adds robustness over problematic mesh features such as holes. The height difference across an edge is thus $heightDiff(\mathbf{e}_k) = |h(f_i) - h(f_j)|$ where f_i and f_j are the faces incident to $\mathbf{e}_k$.

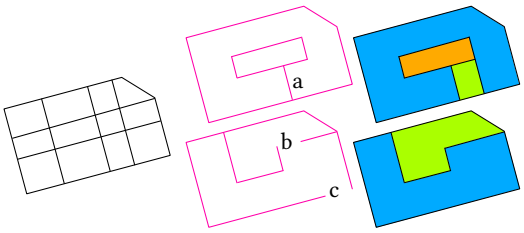


Fig. 7. *Valid footprint-polygons.* Left: A set of edges, $\{s\}$. Center: Two geometrically invalid partitions using those edges caused by self-grazing polygons (a), dangling edges (b), and holes in the boundary (c). Right: Valid footprint-polygons are map-coloring solutions.

Selection variables: The face-merging problem can be reduced to a region- (or map-) coloring problem with adjacent faces of the same color indicating that the faces are implicitly merged. Thus, for each f_i, we assign a selection variable, $\boldsymbol{\gamma}^i$, with length 5. Although four colors are sufficient for map-coloring, we found experimentally that our BIP converges faster with an extra color.
Constraints: The edge-selection variable, s^k, defines if an edge, e_k, lies on a footprint-polygon; usually this is because it lies between faces of different colors. Thus, for all edges, $\mathbf{e}_k$, between two faces f_i and f_j, we require

$$s^k = \textsc{isDifferent}(\boldsymbol{\gamma}^i, \boldsymbol{\gamma}^j),$$

which amounts to a set of variables and constraints as introduced in Section 3.4. Since all other edges, $\mathbf{e}_k$, are at the boundary and must be part of a footprint-polygon, we set their s^k to 1.
Objective terms: In formulating the selection of edges from the tessellation, $\mathbb{G}$, we add penalties for the following conditions: (O_1) if a sweep-edge *is not* selected or a soft-edge *is* selected; and (O_2) if an edge with high height differential is *not* selected

$$\begin{aligned} O_1(\{s^k\}) := & \sum_{\mathbf{e}_k \in \mathbb{G}} 2\|\mathbf{e}_k\|(\neg s^k \wedge \mathit{isSweepEdge}(\mathbf{e}_k)) \\ & + \sum_{\mathbf{e}_k \in \mathbb{G}} \|\mathbf{e}_k\|(s^k \wedge \neg \mathit{isSweepEdge}(\mathbf{e}_k)) \\ O_2(\{s^k\}) := & \sum_{\mathbf{e}_k \in \mathbb{G}} \|\mathbf{e}_k\| \; \mathit{heightDiff}(\mathbf{e}_k) \neg s^k, \end{aligned}$$

where $\mathit{isSweepEdge}(\mathbf{e}_k)$ returns 1 if the edge, $\mathbf{e}_k$, is a sweep-edge, or 0 if it is a soft-edge.

4.1.2 Selecting clean-profiles. The input mesh data are noisy, incomplete, and often contain spurious geometry (e.g., trees or cars). Our goal is to abstract the raw input by assigning a clean-profile from the set, C, to every $\mathbf{e} \in \mathbb{G}$. These assigned profiles guide the footprint-polygon extrusion, implicitly producing a clean and abstracted PE mass model.

Ideally, above each edge, the selected profile closely approximates the mesh geometry. Further, due to stability considerations when modeling with PEs, it is important that edges from adjacent and nearly parallel edges in the same footprint-polygon select the same profile (see Figure 8). Note that this caveat does not require buildings to conform to the Manhattan-world assumption.
Selection variables: For every edge, $\mathbf{e}_k$, we create a profile selection vector, $\boldsymbol{\eta}^k$, to indicate which clean-profile is selected from the global set, C. The length of this vector is the size of the profile set, C, typically 4-80 profiles.
Constraints: We wish clean-profile selections to be equal for parallel adjacent edges within the same footprint-polygon. In other words, two adjacent edges that are nearly parallel can select different profiles *only if* the they belong to different footprint-polygons— i.e., there is at least one separating wall between them.

Thus, for all vertices of the tesselation, $\mathbb{G}$, we create an auxiliary variable for each pair of adjacent and approximately parallel (we use a tolerance of 0.1 radians) edges, $\mathbf{e}_j$ and $\mathbf{e}_k$, as

$$r^{(j,k)} = \textsc{isDifferent}(\boldsymbol{\eta}^j, \boldsymbol{\eta}^k).$$

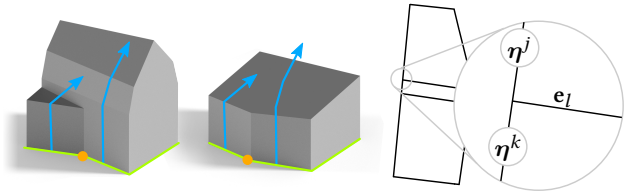


Fig. 8. *Undesirable façade splits.* Left-center: PEs are unstable when different profiles (blue) are selected on nearly parallel edges (green); moving a single point (orange) a short distance creates a very different result. Right: To avoid this situation, the clean-profiles of the adjacent parallel edges (given by the selection vectors $\boldsymbol{\eta}^j$ and $\boldsymbol{\eta}^k$) are constrained to be equal, if the dividing edge is selected ($s^l = 1$).

Because we allow only parallel and adjacent edges to have different profiles ($r^{(j,k)} = 1$) when there is at least one selected edge ($s^l = 1$ for edge $\mathbf{e}_l$) between them at their shared vertex (Figure 8), we require

$$r^{(j,k)} \leq \sum_{\mathbf{e}_l \in \mathit{between}(j,k)} s^l,$$

where $\mathit{between}(j,k)$ denotes the set of edges lying between $\mathbf{e}_j$ and $\mathbf{e}_k$ and sharing a common vertex. We implement $\mathbb{G}$ as a half-edge data structure, which permits direct implementation of the *between*() operator.
Objective term: For each edge, $\mathbf{e}_k$, let the corresponding set of raw-profiles obtained by vertically slicing the input mesh be $\mathcal{R}(\mathbf{e}_k)$. Let the vector $\mathbf{F}_k$ list the error in fitting each clean-profile, $p_c \in C$, to all the raw-profiles, $q \in \mathcal{R}(\mathbf{e}_k)$, along the edge, $\mathbf{e}_k$. This error is measured by the function $d()$, which measures the difference between two profiles (see Section 5.1 for details). Specifically, each element of the vector, F_k^c, is computed for a single clean-profile, $p_c \in C$, over all the edge's raw-profiles as

$$F_k^c = \sum_{q \in \mathcal{R}(\mathbf{e}_k)} d(p_c, q, \min_Y(q), \max_Y(q)).$$

Note that for the above computation, p_c is moved to align with q at height $y = 0$ (i.e., on the sweep-edge). Further, the function $d()$ is evaluated over the raw-profile's height, $[\min_Y(q), \max_Y(q)]$, to match raw-profiles with ends at varying heights to the more complete clean-profile. If there is no raw-profile associated with an edge, we set the assignment cost vector, $\mathbf{F}_k$ to $[-1, 0, \ldots 0]$, i.e., we give a small bonus to selecting the vertical clean-profile. (Note that the -1 favors the default vertical profile in the absence other information.) We can now define an objective term for each edge, $\mathbf{e}_k$, measuring the fit of the selected clean-profile to the supporting edge's raw-profiles,

$$O_3(\{\boldsymbol{\eta}^k\}) := \sum_{\mathbf{e}_k \in \mathbb{G}} \|\mathbf{e}_k\| \mathbf{F}_k \cdot \boldsymbol{\eta}^k.$$

We recall that each internal sweep-edge potentially has two sets (for a shared wall) of raw-profiles associated with it, corresponding to the two adjacent buildings. The above cost is adapted accordingly when a pair of edges is present.

4.1.3 Retargeting building-façades. Street-level imagery of façades contains valuable information about building placement. For example, neighboring buildings may have different materials which provides evidence about their widths, or a change in façade height may advocate splitting a footprint-polygon. However, street-level imagery often does not align with the 3D mesh (or even other images) — both in position and scale. We extend our formulation to include such street-level imagery by observing that solving for alignment and scaling is equivalent to establishing correspondence between the start and end building-façade-points, and the vertices on the boundary of the tessellation.

Specifically, let the set of vertices on the outer boundary of $\mathbb{G}$ be $\mathbb{V}$. We aim to assign every building-façade-point to a vertex, $v \in \mathbb{V}$. Because the error in the building-façade location is of a known maximum distance (approximately 3m in our datasets), we can enumerate the nearby boundary vertices for each building-façade-point. In the process, we aim to minimize both the building-façade-point displacement and the height disparity between the building-façade-based (street-level imagery), and mesh-based, estimates. We note that multiple images may create overlapping building-façades, with each suggesting a corresponding set of façade elements.

Selection variable: We cluster nearby building-façade-points to a group, $\mathbb{C}_i$, with a cluster-representative denoted by $m^i_\star$. For each cluster-representative, we find the nearby boundary vertices in $\mathbb{V}$, denoted as $nearby(m^i_\star)$. We use a selection variable, $\tau^{(i,w)}$, to identify the points in $\mathbb{C}_i$ mapped to vertex v_w.

Objective terms: We introduce three terms: (O_4) to discourage stretch and height disparities between heights extracted from the mesh and those from the street-level imagery; (O_5) to encourage building-façade-points to pick exterior corners of the tessellation; and (O_6) to reduce splitting of footprint-polygons under a building-façade.

First, to minimize stretch and height disparity of the building-façades (see Figure 9), we add

$$O_4(\{\tau^{(i,w)}\}) := \sum_{\forall \mathbb{C}_i} \sum_{m_a \in \mathbb{C}_i} \sum_{w \in nearby(m^i_\star)} \tau^{(i,w)} (distance(v_w, m_a) + |htLeft(m_a) - htLeft(v_w)| + |htRight(m_a) - htRight(v_w)|),$$

where the function *distance*() gives the distance between a boundary vertex and building-façade-point, and *htLeft*() gives the building-façade height or face height (from the street-level imagery or the 3D mesh, respectively), on the left (similarly for *htRight*()), as shown in Figure 9.

It is particularly desirable to assign a building-façade-point to a corner vertex of the tessellation boundary (a subset of $\mathbb{V}$); thus, it receives a reward

$$O_5(\{\tau^{(i,w)}\}) := - \sum_{v_w \in corners} \tau^{(i,w)},$$

where the set *corner* contains all vertices adjacent to two boundary edges of $\mathbb{G}$ that meet at $[\pi/3, 2\pi/3]$.

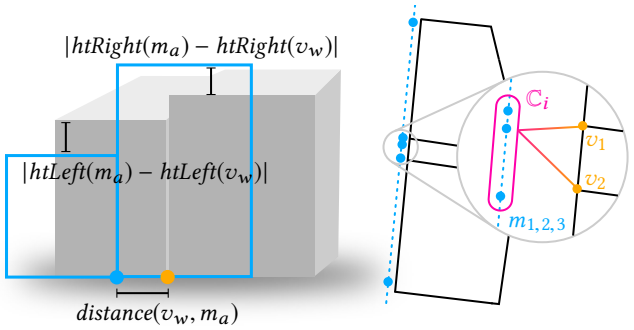
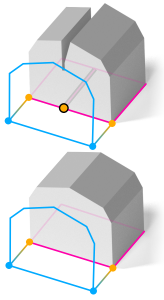


Fig. 9. *Stretch and height disparity.* Left: We evaluate the fit of the building-façade (blue) to the 3D mesh (grey) using stretch and height disparity. Right: The building-façade-points, m_1, m_2, m_3, are grouped into a cluster, $\mathbb{C}_i$, with representative $m^i_\star$. The indicator variable $\gamma^{(i,1)}$ (or $\gamma^{(i,2)}$) denotes which of the points in cluster $\mathbb{C}_i$ are mapped to the boundary vertex, v_1 (or v_2).



Finally, it is undesirable for an edge to be selected that arrives at the tessellation boundary underneath a building-façade (inset: top, blue). Such an edge may unnecessarily split a footprint-polygon (pink). Hence, we penalize the selection of edges, $\mathbf{e}_k$, that approach vertices of the boundary with building-façades, but without selecting building-façade-points. This results in improved integration of the façade boundaries into the mass model (inset, bottom). Specifically, we penalize such a situation as

$$O_6(\{l^k\}) := \sum_{\mathbf{e}_k \in \mathbb{G}} s^k \wedge l^k.$$

Constraints: The auxiliary binary variable, l^k, captures whether a vertex of edge, $\mathbf{e}_k$, is not assigned a building-façade-point, but is covered by a building-façade,

$$l^k = \sum_{v_w \in verts(e_k)} \left(free(v_w) \sum_{\forall \mathbb{C}_i} \tau^{(i,w)} \right) < 1.$$

The above constraint evaluates whether an edge, e_k, has a boundary vertex, $v_w \in verts(e_k)$, which is covered by a building-façade, but is not assigned a building-façade-point by any τ. The function $free(v_w)$ returns 0 if the vertex, v_w, is covered by some building-façade and 1 otherwise.

4.1.4 Objective function. We find a solution that satisfies all the above constraints, while minimizing

$$\min \sum_{i=1}^{6} \alpha_i O_i$$

over the variables $\{\boldsymbol{\gamma}^i\}$, $\{\boldsymbol{\eta}^k\}$, $\{\boldsymbol{\tau}^{(i,k)}\}$, and the associated auxiliary variables. In our results, we used $\alpha_1 = 10$, $\alpha_2 = 1$, $\alpha_3 = 0.01$, $\alpha_4 = 1$, and $\alpha_5 = \alpha_6 = 0.1 \sum_{\mathbf{e}_k \in \mathbb{G}} \|\mathbf{e}_k\|$.

Fig. 10. *Overlapping building-façade elements.* An urban block is typically covered by multiple overlapping façade images, giving repeated bounding rectangles for many elements, such as windows (turquoise), shops (pink), balconies (orange), doors (green), mouldings (dark blue), and building-façade boundaries (light blue). The left-most visible facade of Figure. 1 was reconstructed from these overlapping elements.

4.2 Creating the Structured Model

Given a *solution* to the above optimization, we can generate the geometry for the final structured model.

Starting from the ground plane tessellation, $\mathbb{G}$, and the region coloring $\{\boldsymbol{\gamma}^i\}$, we merge neighboring faces to which the solution assigns the same color. The resulting 2D polygons form the footprint-polygons of the final mass models. For every edge in each footprint-polygon, the solution contains a clean-profile from the set C, in $\{\boldsymbol{\eta}^k\}$. Procedural extrusions [Kelly and Wonka 2011] lift each footprint-polygon using the selected clean-profiles to create a building's mass model. During this extrusion, we cap the PE mesh at the average mesh height (sampled by $h(x, z)$ over the footprint-polygon) to stop runaway geometry and to create flat roofs. An exception is when the PE horizontal cross-section area is decreasing rapidly at this height, in which case we assume that the roof is pointed. We cap pointed roofs at a higher level given by the average raw-profile height around the footprint-polygon boundary. We classify the surfaces obtained via PE as walls or roofs using the local normals.

The optimization solution assigns the building-façades to portions of the mass-model. This correspondence between building-façade-points and vertices of the footprint-polygons is given by $\{\boldsymbol{\tau}\}$; from this, we can position the building-façades over the mass models.

The building-façade's points are found from image features. One, or both, points may be missing because they lie outside the image. If both points are present, we translate and scale the building-façade to align its building-façade-points with the corresponding footprint vertices. If only one point is present, we simply translate it to align with the found vertex. In the case of no points, the building-façade is aligned using estimated Google StreetView (GSV) pose data.

In this manner, multiple building-façades can be positioned over the same section of the mass model, giving us multiple position estimates for façade elements (doors, windows, balconies etc., see Figure 10). Further, because these elements have been estimated from street-level imagery, they contain noise and omissions.

In the following, we explain our fusion and regularization process for window elements, while other element classes (doors and balconies, etc.) are treated similarly. We adopt a simple mean-shift [Fukunaga and Hostetler 1975] approach; at each iteration, we apply a step of 0.2× the mean-shift vector to all window rectangles for a variety of parameterizations. Namely: (i) absolute position of the left, right, top, and bottom of the rectangle (to align windows with themselves and others in a grid); (ii) width and height (to maintain the shape of the windows in subsequent iterations); and (iii) spacing between adjacent windows to the left, right, top and bottom (to encourage uniform spacing between windows). After the mean-shift has converged (we use 30 iterations), we frequently have multiple rectangles associated with each window. Such rectangles are merged if the overlap is more than 50%; otherwise, the smallest rectangles are discarded. Element rectangles are also discarded if they occur in less than half of the street-level images that cover them.

These element rectangles are added to the mass model using simple *parametric models* for each type of element, such as windows, doors, window-sills, cornices, moldings, and balconies. These are parameterized to the found dimensions, and windows or doors are recessed into the mass model façade. As an exception, windows that lie on a mass model surface that is classified as a roof, or between surfaces with different normals, are added as dormer windows.

Finally, we color the mass model polygons classified as *wall* using the information extracted from the street-level imagery, and those classified as *roof* using optional satellite image information. Figure 1 shows such a resulting structured model.

5 IMPLEMENTATION DETAILS

5.1 Extracting Sweep-edges and Profiles

We now describe the profile analysis of the 3D mesh and GIS footprints. First, we align the mesh with the GIS footprint boundary. Then, we create and cluster *horizontal-lines* (Figure 11b, c) to find the *prominent-faces* of the building-block (Figure 11d). Each such face is used to compute a *sweep-edge* on the ground plane, along which we extract vertical *raw-profiles* from the mesh (Figure 11e). The profiles are processed to create a small, yet representative, set of *clean-profiles*, C.

First, we align the mesh to the GIS footprints using the GPS position associated with the mesh. We use the GIS footprint boundary to discard mesh geometry more than a street-lane width away (typically 4m) from the building-block of interest.

We found horizontal-lines to be good indicators of predominant directions in architectural meshes; they also support the strong horizontal edges that are characteristic of PEs. To find such lines, we slice the mesh horizontally (we used 20cm intervals), and simplify each such slice using polyline fitting (Figure 11a). Because the mesh may have holes and noise, we use the directions in the GIS footprints to regularize the line fitting (Figure 11b). Specifically, if lines are within 20° of the closest GIS edge, they are rotated to match the GIS line's orientation.

We now cluster the fitted horizontal-lines based on their orientation to identify prominent-faces of the building-block (e.g., a south-facing wall). The seed of the cluster is the longest horizontal-line (Figure 11c, bold). From this seed-line we progressively build the cluster by adding neighboring lines (from slices above and below) in a "floodfill" fashion, ensuring that each line's orientation matches that of the seed-line (within 20°). Such a cluster of lines defines a *prominent-face* over the mesh. We continue to create prominent-faces by taking the next longest unused horizontal-line as a seed and repeating the floodfill. We discard any prominent-faces that cover a small area of the mesh; we use a threshold of approximately $30m^2$, which balances preserving detail with removing noise.

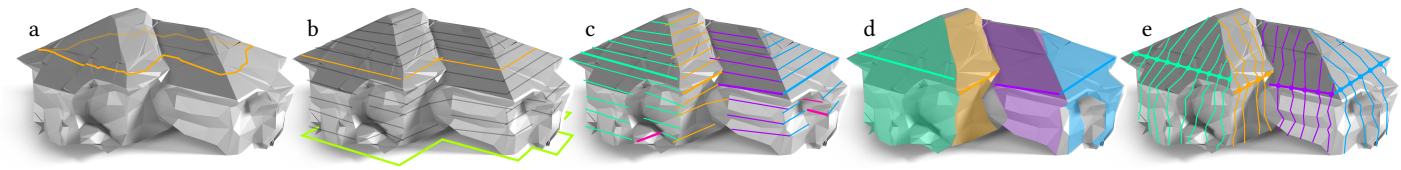


Fig. 11. *Sweep-edges and profile analysis.* A horizontal slice of the mesh (a, orange), has polylines fitted to it (b) and is regularized by the GIS information (b, green). These are clustered from the seed-lines (c, bold lines), and the associated prominent-faces (d), which can be used to find the raw-profiles (e).

The prominent-faces are now sampled to obtain profiles (Figure 11d). A *profile* is a weakly y-monotone polychain (i.e., every point is greater, or equal, in height to every preceding point). This monotonic property is required by PEs, which we observe is satisfied by a large majority of building types. We continue to extract a set of raw-profiles directly from the 3D mesh; the mesh is sliced perpendicularly to the seed-line's direction at regular intervals (20cm). Nearly horizontal mesh faces (with a normal approximately 5° from vertical), or those not associated with the prominent-face are ignored. We create a raw-profile by traversing a portion of the slice, starting at the closest point on the slice to the prominent-face's seed-line. The traversal takes place upwards and downwards, selecting monotonic line-segments from the slice to add to the profile. It jumps over small gaps and non-monotonic sections of the slice by searching for the next point in a small locale (approximately 2m).

We now use the raw-profiles to find a smaller, yet representative, set of clean-profiles, C. We first cluster the raw-profiles along each sweep-edge using profile distance. Given two monotone profiles, p_i and p_j, we define the profile difference at a height, y, as

$$\delta(p_i, p_j, y) = \begin{cases} \sqrt{(x(p_i,y) - x(p_j,y))^2 + 4(\angle(p_i,y) - \angle(p_j,y))^2} \\ \qquad \text{if } p_i \text{ and } p_j \text{ are defined at height y,} \\ 10 \qquad \text{otherwise.} \end{cases}$$

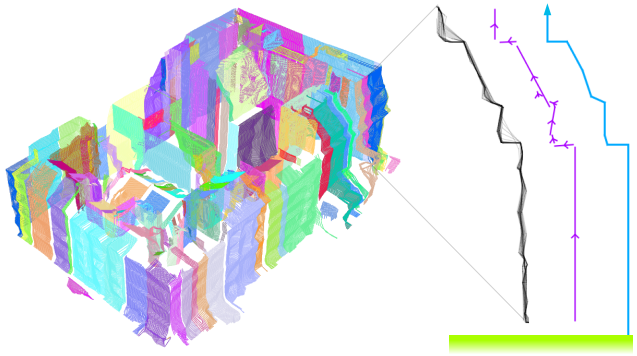


Fig. 12. *Raw- and clean-profiles.* Left: Each color represents a cluster of adjacent and similar raw-profiles from Figure 1. Right: A cluster of raw-profiles (grey) has line segments fitted to it (purple) and is finally regularized to yield a clean-profile (blue).

where $x(p_i, y)$ and $\angle(p_i, y)$ are, respectively, the x-position and angle (in radians), of profile p_i at height y. When the profiles range between heights y_l and y_u, the cumulative distance function is then the mean horizontal distance between the profiles discretized over the vertical range $[y_l, y_u]$ as

$$d(p_i, p_j, y_l, y_u) := \sum_{y \in [y_l, y_u]} \delta(p_i, p_j, y)/(y_u - y_l).$$

The raw-profiles are clustered by examining consecutive profiles along each sweep-edge, starting a new cluster whenever

$$d(p_{last}, p_{next}, 0, max_Y(p_{last}, p_{next})) > t,$$

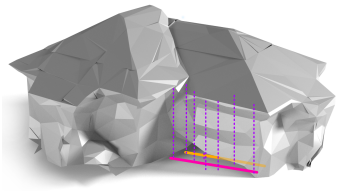
where t is a threshold value and $\max_Y(p_i, p_j)$ is the maximum height of profiles p_i and p_j. Small clusters with fewer than five profiles are discarded. Empirically, we find that forming clusters from such contiguous portions of sweep-edges gave better results than techniques such as spectral clustering, because it prioritizes the strong spatial-correlation between adjacent raw-profiles. Examples of such clusters are shown in Figure 12-left.

To create a simplified *clean-profile* from each cluster of raw-profiles, we fit a set of line segments (Figure 12-right). Using strong architectural priors, we regularize these lines into a clean-profile. Because of the low resolution of our input meshes, we found we could aid regularization by requiring the profiles to be both vertically *and* horizontally monotonic (note that PEs require only that the profiles be vertically monotonic).

We used the following rules to create the clean-profiles (see Figure 12-right): (i) lines that are nearly horizontal or vertical are snapped to these orientations. Near the ground, this snapping is very aggressive to mitigate the effect of occluders; (ii) lines that do not form part of vertically and horizontally monotonic profiles are either removed or sliced so that they do; (iii) lines that are near the ground are extended to the ground; and, finally, (iv) if two adjacent lines could be extended to intersect within 2m of an end of both lines, we extend the lines to this intersection. We add the resulting clean-profile to the profile set, C.

A large number of clean-profiles in C are computationally expensive in the optimization stage (Section 4). Hence, we aggressively reduce them by: (i) removing pairs of similar profiles from the pool using d (we used $d() < 1$); (ii) discarding any profile that is not preferred by some cluster of raw profiles, and (iii) replacing all simple vertical profiles with a single vertical profile at the start of C.

Fig. 13. *Training data and façade classification.* Top-left: The ground truth used to for the 'Façade' and 'Window' labels. Remainder: The source image (top-right) used to compare a model trained to recognize a single set of disjoint labels (bottom-left) with one trained to recognize independent sets of labels for each type of feature with edge labels (bottom-right). Each model was trained for 150 Epochs. The second option leads to crisper features.



Finally, for each prominent-face, we compute a *sweep-edge*. Sweep-edges represent potential wall positions over the ground plane, and, along with suitable vertical clean-profiles, create the 3D mass models. We find a sweep-edge by projecting the seed-line of each prominent-face onto the ground plane (inset; orange line), and offsetting it to lie close to the start of the profiles (inset; pink line). This offset is necessary because the found seed-line may not be on the structure's wall. The offset is the mean horizontal distance from the seed-line to the bottom of the raw-profiles. This set of sweep-edges, $\mathcal{S}$, represents the potential wall-positions.

5.2 Acquiring Street-level Imagery

We use street-level imagery from Google StreetView (GSV) to estimate the locations of façade elements such as windows, balconies, doors, and moldings, as well as the locations of façade boundaries. Unprocessed GSV images are 360° panoramas including approximate pose data (position and orientation of the rig used to capture the images) that are estimated using GPS and a variety of additional techniques described by Anguelov et al. [2010]. Based on the GSV pose information and GIS footprints, we project the GSV panorama images onto the expected façade plane to obtain a (roughly) rectified *projected image*.

These projected images are generated at a resolution of 40 pixels / meter. We crop the images to a fixed horizontal field of view of 120°. This is centered on the projection of the paranorama center onto the façade plane. We use a fixed field of view to avoid distortion caused by projecting the panorama at extreme angles. This results in more than one overlapping image of each façade and many images containing only a portion of a façade. We note that some façades have no GSV images because of legal and physical constraints on photography. A typical example of missing imagery is the private courtyards found in the center of many European city blocks. Next, we describe how to find façade features in the projected images.

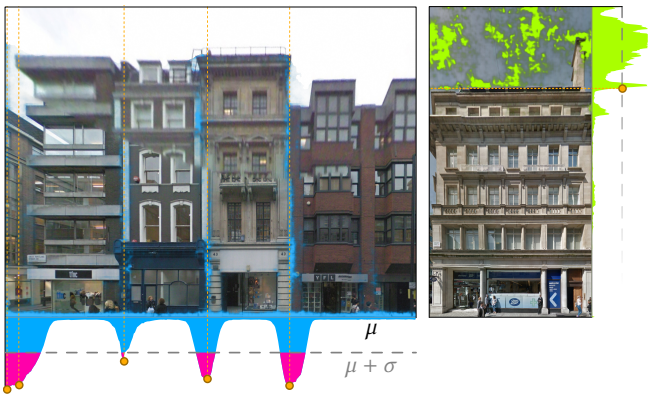


Fig. 14. *Finding façade extents.* Left: We split images with multiple façades based on the peaks of the vertical sums of the façade 'Edge' scores that are output by the segmenter (superimposed in blue over the image); façades are split at the highest point of each interval where the projection's value is more than one standard deviation (σ) above its mean (μ). Right: The integral of the detected 'Sky' label (green) is used with a threshold to identify the top of the façade.

5.3 Analyzing Street-level Imagery

Starting from input street-level imagery, our goal is to detect each façade's location and dimension, and its building elements (e.g., windows, balconies, etc.). A *building-façade* records this information for one image and one estimated façade; we refer to the set of building-façades as $\mathcal{B}$.

In practice, we found the GSV pose estimates to be insufficient to produce projected street-level imagery that is sufficiently aligned with GIS data. In the example of London, we observed overlaid GSV imagery to deviate from GIS building footprints by nearly 3m on the façade plane, or 5° in GSV panoramas. Therefore, a pre-processing step removes parts of the images that are unlikely to be part of a façade and then rectifies each image. The unwanted features are segmented and masked-out using the *Bayesian SEGNET* CNN [Badrinarayanan et al. 2017; Kendall et al. 2015]. This network was trained on urban street scenes using CamVid data [Brostow et al. 2008] and then refined using CityScapes data [Cordts et al. 2016] to identify parts of images that are likely to have façade features. We then rectify based on the edges within that region using the method proposed by Affara et al. [2016].

Next, we identify the façade elements within these rectified images. We refine the probabilistic Bayesian SEGNET architecture to segment a set of labels for architectural façade element features using the CMP Facade dataset [Tylecek 2012], the dataset used by Affara et al. [2016], and an additional dataset of 800 facades that we annotated directly from GSV images of London, Oviedo, and New York. We use this *SEGNET-FACADE* model (available at: https://github.com/jfemiani/facade-segmentation) to assign per-pixel probabilities to the images for each feature class.

Traditional segmentation approaches, including SEGNET, assign a single label to each pixel in an image. In contrast, we treat façade segmentation as a number of separate labeling tasks, one for each class of façade element (window, shop, balcony, molding, door etc.),

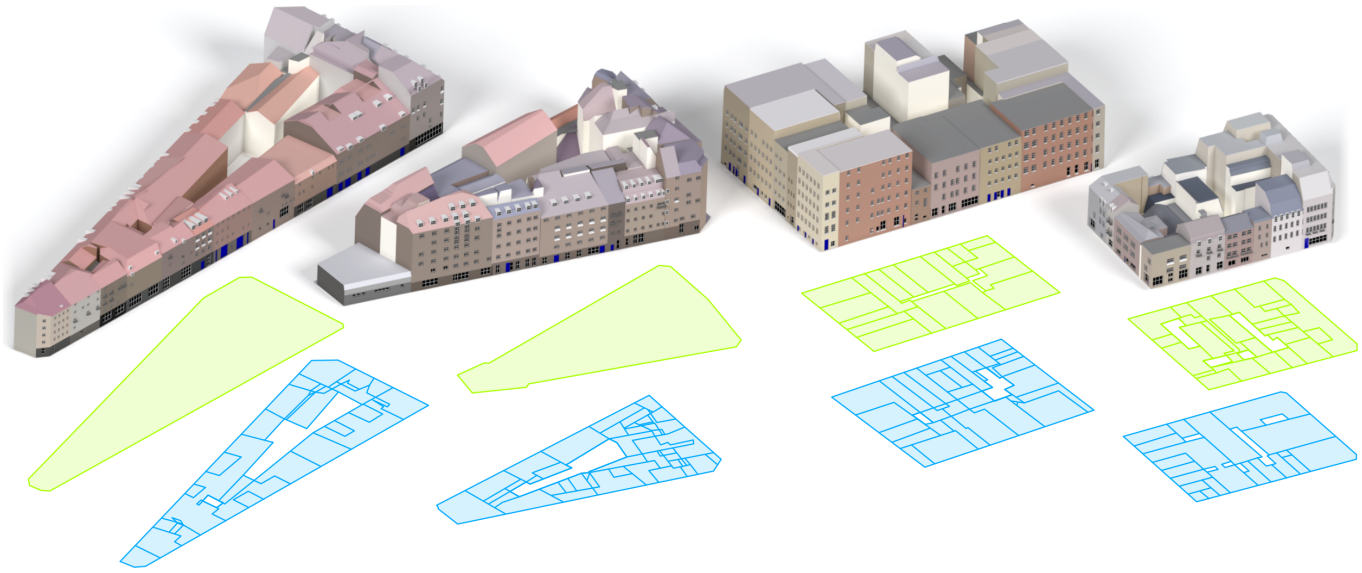


Fig. 15. *Oviedo, Manhattan, and London.* City blocks from Oviedo (left, center-left), Manhattan (center-right) and London: Little Portland Street (right, as Figure 1). Top: Our results. Bottom: input GIS footprints (green) and optimization output floorplans (blue). See also Table 1.

and one for the façade extent itself. Each task assigns one of four labels to each pixel; 'Negative', 'Positive', 'Unspecified' (which is ignored), or 'Edge'. The 'Edge' label is automatically assigned to a thin region (6 pixels spanning an estimated 15cm) around the edge of each feature, with the exception of vertical façade edges, where the 'Edge' label is assigned to a wider region (15 pixels wide, spanning approximately 38cm). Using a separate 'Edge' label ensures more weight is given to the training-loss in these pixels due to median frequency balancing [Eigen and Fergus 2015]. Empirically, these improvements result in sharper features, as shown in Figure 13, which is useful for isolating individual feature instances. The CNN processes images at a resolution of 512×512 pixels. We rescale all images to a height of 512 pixels and crop the widths. During inference, several horizontal tiles are used to cover an image.

The GSV images often contain multiple façades, and it is important to separate them into different individual building-façades for the optimization. At the inference stage, we sum each pixel column's Bayesian SEGNET-FACADE scores for the 'Edge' label. This one-dimensional signal peaks at each façade boundary. The signal is dilated by 60 pixels (1.5m) in order to merge the dual-peaks that can occur if the street-level imagery is imperfectly rectified, or if there are stitching artifacts (see Figure 14). We extract peaks as local maxima that are more than one standard deviation above the mean of the dilated signal (see Figure 14-left). Each façade image is split at these peaks to produce *building-façades*. For each building-façade, we produce axis-aligned bounding boxes of all features as shown in Figure 14-right. In order to estimate the height of each building-façade, we use the original SEGNET to label pixels as 'Sky'. The 85^{th} percentile of the scores at each pixel-row forms a one dimensional sky signal (see the green region of Figure 14-right). The top of the façade is the lowest point where the sky signal crosses 50%. These width and height estimates are assigned to each building-façade and used in the optimization stage. Because we know the location of the façade image-plane in $\mathbb{R}^3$, the building-façade has an estimated 3D position, as do the associated features.

5.3.1 Training and Evaluation. We trained SEGNET-FACADE on 80% (1173 images) of the data we collected, an additional 20% (293 images) were used to evaluate the precision, recall, and F_1-scores of our approach. SEGNET-FACADE obtained a per-pixel precision of 96%, recall of 69%, and an F_1 of 0.80. By comparison SEGNET trained on the same data obtained a per-pixel precision of 73%, recall of 62% and an F_1 of 0.67. We also evaluated per-object precision by defining a successful match between objects as an intersection-over-union over 50%. The per-object scores gave a precision of 88%, recall of 68%, and an F_1 score of 0.77. We consider these to be useful results as many of the façade images were collected "in the wild" from GSV and imperfectly rectified. In comparison, SEGNET acheived precision of 36% and recall of 28%, with an F_1 of 0.32. The recent method of Affara et al. [2016] had a per-object precision of 85%, recall of 52%, and an F_1 score of 0.64 on the same data.

5.3.2 Collecting color estimates. Although a façade may contain a variety of texture and color patterns, we limit ourselves to a single color; additional color variation comes from the inclusion of façade elements with fixed colors, such as windows, molding, cornices, sills, and balconies. To estimate the color of the walls, we mask out all regions that have been identified as any other feature and estimate the mode color in the remaining pixels. Specifically, we use the *Lab* color space and select 50 colors randomly from the (unmasked) façade. The color with the most matches is selected as representative of the façade. Optionally, a separate color can be used for the ground floor and for the higher stories. In this case, we estimate the ground floor height by finding the highest row in the image with the 'Shop' label.

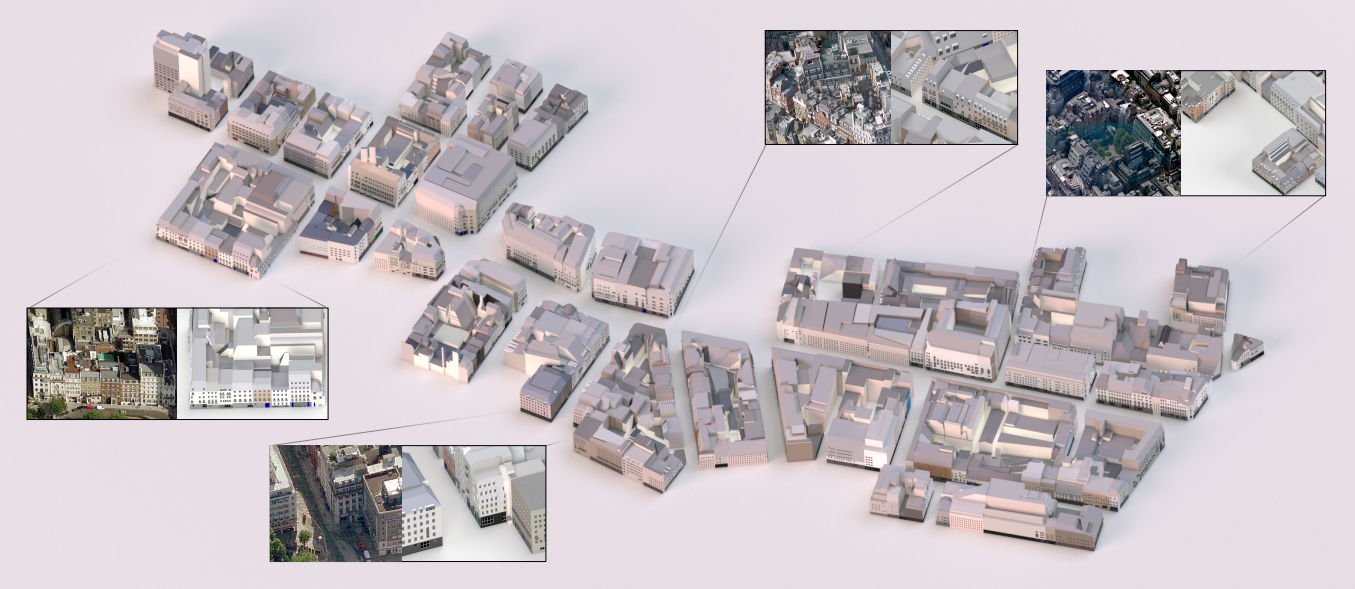


Fig. 16. *London: Oxford Circus blocks.* Structured urban reconstruction spanning 37 blocks and 1,011 buildings.

6 RESULTS

We implemented the proposed framework using Java and Python; the sourcecode is available online at the project page (http://geometry.cs.ucl.ac.uk/projects/2017/bigsur). We used Gurobi [Gurobi 2016] for binary integer programming and Caffe [Jia et al. 2014] for the CNN-based classification. The timings were recorded on an i7-7700K desktop (with the exception of the Oxford Circus example).

We demonstrate our framework on building blocks from different cities: Detroit (see Figure 17), Manhattan and Oviedo (see Figure 15), and London (see Figure 1 for Little Portland Street and Figure 16 for Oxford Circus). We selected building blocks to show a variety of inputs, from free standing single-family houses in Detroit to dense urban areas in the other three selected cities. We selected cities with

Table 1. Details for Figure 15. Values are given for location, number of clean profiles ($|C|$) and sweep edges ($|\mathcal{S}|$), binary variables (vars) and constraints (constr), number of output footprints (fp), and the solve times.

Fig:col	location (lat,long)	$\|C\|$	$\|\mathcal{S}\|$	vars	constr	fp out	solve time
15:1	43.36635, -5.83256	75	61	32,242	73,193	34	15h
15:2	43.36584, -5.83189	73	56	74,694	148,945	38	5h
15:3	40.72191, -74.00131	46	30	23,172	49,941	37	4h
1:1 15:4	51.51724, -0.14199	58	60	45,249	88,171	28	4h

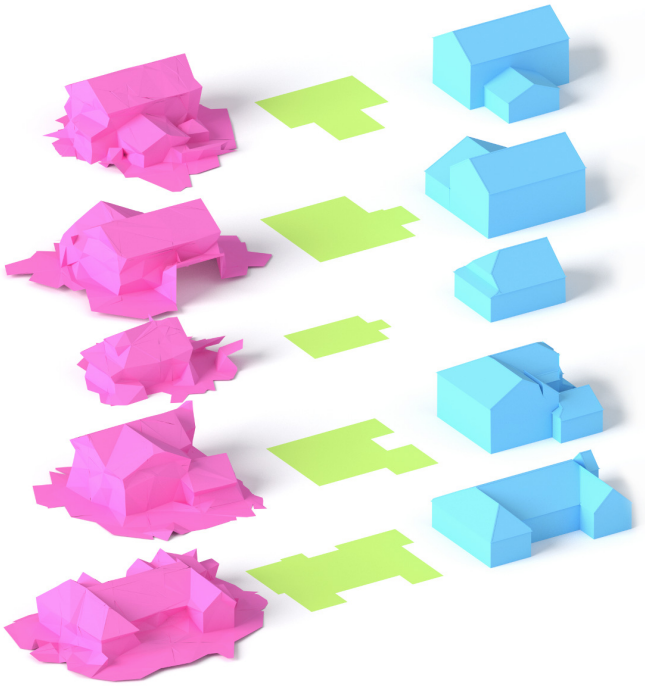


Fig. 17. *Detroit.* Without building-façades, our technique exhibits strong architectural regularization with the coarse mesh (pink) and GIS footprint (green, no interior edges) as inputs. See Table 2 for details.

Table 2. Details for Figure 17, columns as in Table 1.

Fig:row	location (lat,long)	$\|C\|$	$\|S\|$	vars	solve time
17:1	42.38458, -82.95086	9	5	196	0.01s
17:2	42.38458, -82.95084	8	7	657	0.05s
17:3	42.38587, -82.95165	6	4	165	0.00s
17:4	42.38614 -82.95125	23	13	1,799	2.92s
17:5	42.38350, -82.94954	37	14	1,494	0.3s

accessible mesh and GIS data. In our experiments, we found most of the parameters to be stable when the input data quality remained consistent. Typically, we adjusted two parameters before running the optimization: the thresholds for the creation of $\mathbb{G}$ and the mesh area for ignoring small clusters of horizontal lines. These parameter adjustments are relatively interactive because they occur before the slow BIP optimization.

6.1 Timings

The computation times are dominated by the time it takes to compute a solution to the binary integer program. We list details of this optimization for selected blocks in Tables 1 and 2. Other components that contribute to the runtime are image processing to extract building-façades (about 45 seconds per image), mesh processing to extract sweep-edges and clean-profiles (less than 20 seconds per block), grid-based regularization of façade elements (less than 3 seconds per façade), basic mass model construction (less than 10 seconds per block), and façade element insertion into the mass models (less than 10 seconds per block).

6.2 Comparison

We compared our work to other related algorithms in Figure 19. As there exists no competing work to fuse multiple data sources, we limited our comparison to the processing of mass models. Therefore, we did not use GIS footprints or building-façades as input to any of the algorithms for this comparison; we used only the polygon soup meshes. To select competing work, we limited our choices to methods that had sourcecode available or where the authors helped us to generate results. The first method in our comparison is Poisson reconstruction [Kazhdan et al. 2006], which can fill some smaller holes in the input, but the output looks similar to the input. Fitting a polygonal model using the Manhattan-world assumption [Li et al. 2016] works well when the geometry conforms to such an assumption. However, we can see that over sloped roofs and within a larger block of buildings, the surface orientations vary too much, allowing the algorithm to produce good results on only one of the three inputs. Finally, we compare our method to structure-aware mesh decimation [Salinas et al. 2015], which also produces good results, but only a part of the model is simplified.

6.3 Little Portland Street

Finally, we also provide results for a larger area in London consisting of 37 building blocks and 1,011 buildings (see Fig. 16). We used 738 images to find 2,716 building-façades giving rise to 19,377 detected features. We used a fixed computational budget of 1 hr for small blocks and 4 hrs for large blocks; the optimization returns the best solution found within the given time. A 40 core ($10 \times$ E5-2630) server was used for this example.

6.4 Limitations

Our system suffers from a few limitations. The PE representation of our mass models uses straight-line segments for footprint-polygons and profiles, so we cannot correctly capture freeform buildings (e.g., buildings with a curved front or requiring a curved profile as in Figure 18). In addition, our aggressive profile processing has the consequence that overhanging structures cannot be represented (e.g., bridges or balconies). Another source of error is misclassifications of façade imagery. This is particularly the case when our classifier encounters datasets with building styles for which it has not been trained. We found datasets from certain European cities to be particularly challenging as the street-level imagery had to be obtained from narrow streets and alleys, resulting in strong perspective distortions. Other reasons for low accuracy classification results are very tall buildings, untrained features (e.g., fire escapes, buses, statues, etc.), or recessed floors that are not visible from street-level imagery. While we expect that our classification results will continue to improve with access to more annotated training data, in the interim, allowing the user to correct mistakes would be a good alternative. Another observed failure case occurs when roof gutters do not align to detected building-façade boundaries, as our optimization assumes such situations are noisy data. Finally, our core

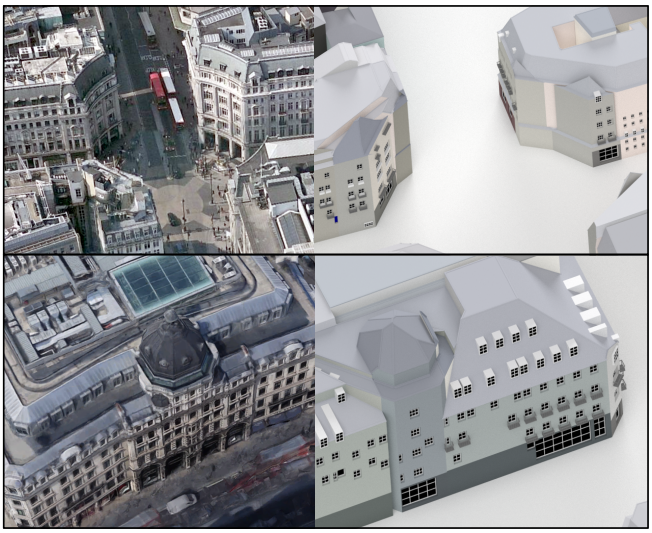


Fig. 18. *Limitations.* (Top) Curved façades can become over-fragmented during sweep-line fitting and then adversely affect the street-level imagery analysis stage, resulting in missed building-façade elements. (Bottom) Another limitation is handling buildings with curved profiles.

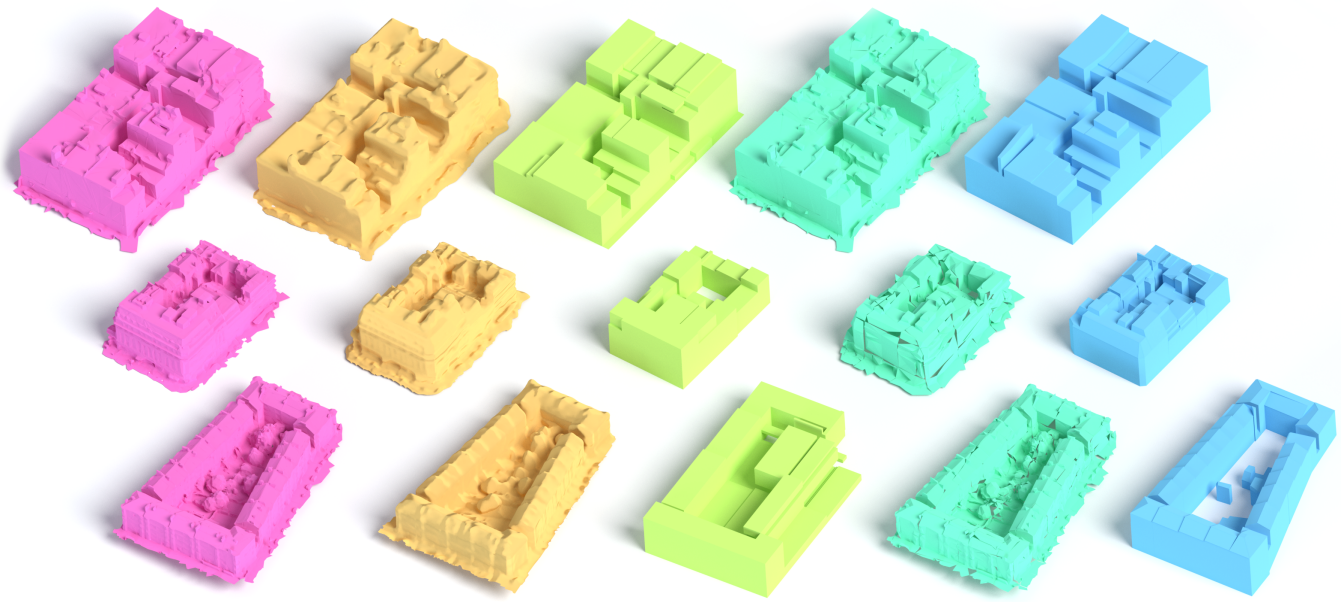


Fig. 19. *Comparison.* Columns left to right, input polygon soup mesh, Poisson reconstruction [Kazhdan et al. 2006], Manhattan box fitting [Li et al. 2016], Structure-Aware Mesh Decimation [Salinas et al. 2015] and our technique (without GIS footprints or building-façade inputs).

optimization relies on a BIP solver that globally combines the input data sets. This prevents us from developing an interactive system because the resulting optimization can run for multiple hours for larger city blocks. However, because the actual coupling is at the city-block level, the problem does not amplify with increasing city size as long as the complexity of the city blocks remains constant.

7 CONCLUSION

We present a system to fuse partial and heterogeneous sources of data, specifically building footprints from GIS databases, polygonal meshes (polygon soup), and street-level imagery, to produce plausible structured models for densely-built building-blocks. Technically, we achieve this by formulating a binary integer program that *simultaneously* considers how to partition the ground plane, assign profiles, and position building-façades. In the process, we globally balance information from incomplete and inconsistent input data to produce a semantically consistent structured model. We evaluated our system on large scale datasets, spanning multiple urban blocks, to produce semantic results at a scale and quality not previously possible using state-of-the-art automated workflows. Incidentally, we introduced a new CNN for detecting façade elements (e.g., windows, doors, etc.) on real-world images, and a mesh processing framework to decompose architectural meshes into footprints and profiles.

Our work opens up several future research directions. As an immediate next step, we would like to evaluate our CNN on other city datasets, and collect additional training data (i.e., labels) on façade images from a wider range of cities to improve classification accuracy. Another interesting direction is to develop a semi-automatic system to allow users to edit inaccurate footprints, profiles, building-façades, or façade elements, to improve the output quality. For example, the user can mark a few smaller features, such as fire-escapes or air-conditioning units, which can then be used to refine city-specific feature detectors. In the longer-term, we envision a two-stage dynamic city-modeling tool, where a few city blocks are initially reconstructed using our proposed system. Once the models are approved by the user, the structured model can be used to obtain a style description of buildings in the city. Such a description can then be used for wider-scale data integration, allowing us to handle large areas of missing data. Thus, the first round of results would act as a prior to synthesize missing information. This workflow would make it feasible to rapidly produce high-quality structured models of entire cities.

ACKNOWLEDGEMENTS

We would like to thank the many people who contributed to this paper; the reviewers, image labellers, and others who read manuscripts, each made valuable contributions. In particular, we thank Florent Lafarge, Pierre Alliez, Pascal Müller, and Lama Affara for providing us with comparisons, software, and sourcecode, as well as Virginia Unkefer, Robin Roussel, Carlo Innamorati, and Aron Monszpart for their feedback. This work was supported by the ERC Starting Grant (SmartGeometry StG-2013-335373), KAUST-UCL grant (OSR-2015-CCF-2533), the KAUST Office of Sponsored Research (award No. OCRF-2014-CGR3-62140401), the Salt River Project Agricultural Improvement and Power District Cooperative Agreement No. 12061288, and the Visual Computing Center (VCC) at KAUST.

A REPRESENTING BOOLEAN OPERATORS IN A BIP

In this appendix, we note that arbitrary Boolean relationships ($\wedge$, $\vee$, $\oplus$, $\neg$, = etc.) can be encoded as IP constraints with additional variables and constraints (see [Chinneck 2008]); such variables are omitted from the main text, but some examples are given in Table 3. Modern IP solvers [Gurobi 2016] are very efficient at solving such trivially constrained sets of variables. Finally, we recall that the logical disjuction of a binary selection vector,

$$r = \chi_1 \vee \cdots \vee \chi_n,$$

can be more efficiently implemented as a summation, given that only one element will take the value 1, as

$$r = \sum_{i=1}^{n} \chi_i.$$

Table 3. Expressing Boolean operations in a BIP.

expression	$c = a \wedge b$	$c = a \oplus b$	$c = a \vee b$
BIP encoding	$c \geq a + b - 1$ $c \leq a$ $c \leq b$,	$c \leq a + b$ $c \geq a - b$ $c \geq b - a$ $c \leq 2 - a - b$	$c \leq a + b$ $c \geq a$ $c \geq b$

B AVOIDING BAD GEOMETRY

The ground tessellation, $\mathbb{G}$, is created by a variety of data sources. Hence, it can contain unlikely combinations of edge selections that we wish to avoid. For example, edges that are parallel, and in close proximity with one another, may create skinny footprint-polygons, while pairs of edges with a small angle between them may produce pointed polygons. Such details are unarchitectural, and we can optionally add a term to our optimization that penalizes undesirable pairs of edges within a polygon (this term was used in the Little Portland Street example shown in Figure 1).

We find pairs of edges within each face that we wish to penalize, $bad(\mathbb{G})$. This set contains pairs of edges that are approximately parallel, and less than 2.5m apart, or are adjacent with an angle less than 30° (pairs of such lines are shown in pink and blue in the above inset). Entries from this set can be discouraged by only selecting one edge from each pair; we model such a penalty term as

$$O_7(\{s^k\}) := \sum_{(\mathbf{e}_i, \mathbf{e}_j) \in bad(\mathbb{G})} s^i \wedge s^j$$

with a large weight of $\alpha_7 = 0.5 \sum_{\mathbf{e}_k \in \mathbb{G}} \|\mathbf{e}_k\|$.

REFERENCES

Lama Affara, Liangliang Nan, Bernard Ghanem, and Peter Wonka. 2016. Large Scale Asset Extraction for Urban Images. *ECCV* (2016), 437–452.

Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. 1996. A novel type of skeleton for polygons. In *The Journal of Universal Computer Science*. Springer, 752–761.

Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. 2010. Google street view: Capturing the world at street level. *Computer* 43, 6 (2010), 32–38.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE TPAMI* (2017).

Therese Biedl, Stefan Huber, and Peter Palfrader. 2016. Planar matchings for weighted straight skeletons. *International Journal of Computational Geometry & Applications* 26, 03n04 (2016), 211–229.

Claus Brenner. 2005. Building reconstruction from images and laser scanning. *International Journal of Applied Earth Observation and Geoinformation* 6, 3 (2005), 187–198.

Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. 2008. Segmentation and Recognition Using Structure from Motion Point Clouds. *ECCV* (2008), 44–57.

Duygu Ceylan, Minh Dang, Niloy J. Mitra, Boris Neubert, and Mark Pauly. 2016. Discovering Structured Variations Via Template Matching. *CGF* (01 2016).

Duygu Ceylan, Niloy J. Mitra, Youyi Zheng, and Mark Pauly. 2013. Coupled Structure-from-Motion and 3D Symmetry Detection for Urban Facades. *ACM TOG* (2013), 15.

John W. Chinneck. 2008. *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*. Springer.

Andrea Cohen, Alexander G Schwing, and Marc Pollefeys. 2014. Efficient structured parsing of facades using dynamic programming. *IEEE CVPR* (2014), 3206–3213.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE CVPR*.

Dengxin Dai, Mukta Prasad, Gerhard Schmitt, and Luc Van Gool. 2012. Learning domain knowledge for facade labelling. *ECCV* (2012), 710–723.

Paul E Debevec, Camillo J Taylor, and Jitendra Malik. 1996. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. *ACM SIGGRAPH* (1996), 11–20.

Anthony R Dick, Philip HS Torr, and Roberto Cipolla. 2004. Modelling and interpretation of architecture from several images. *IJCV* 60, 2 (2004), 111–134.

David Eigen and Rob Fergus. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *IEEE ICCV* (2015), 2650–2658.

David Eppstein and Jeff Erickson. 1999. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete & Computational Geometry* 22, 4 (1999), 569–592.

Tian Fang, Zhexi Wang, Honghui Zhang, and Long Quan. 2013. Image-based modeling of unwrappable facades. *IEEE TVCG* 19, 10 (2013), 1720–1731.

K. Fukunaga and L. Hostetler. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE TIT* 21, 1 (January 1975), 32–40.

Yasutaka Furukawa and Jean Ponce. 2010. Accurate, dense, and robust multiview stereopsis. *IEEE PAMI* 32, 8 (2010), 1362–1376.

Ignacio Garcia-Dorado, Ilke Demir, and Daniel G Aliaga. 2013. Automatic urban modeling using volumetric reconstruction with surface graph cuts. *Computers & Graphics* 37, 7 (2013), 896–910.

Aleksey Golovinskiy, Vladimir G Kim, and Thomas Funkhouser. 2009. Shape-based recognition of 3D point clouds in urban environments. *IEEE ICCV* (2009), 2154–2161.

Gurobi. 2016. Gurobi Optimizer Reference Manual. (2016). http://www.gurobi.com

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).

Haiyong Jiang, Liangliang Nan, Dong-Ming Yan, Weiming Dong, Xiaopeng Zhang, and Peter Wonka. 2016. Automatic constraint detection for 2D layout regularization. *IEEE TVCG* 22, 8 (2016), 1933–1944.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. *SGP* (2006), 61–70.

Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM TOG* 32, 3 (2013), 29.

Tom Kelly and Peter Wonka. 2011. Interactive architectural modeling with procedural extrusions. *ACM TOG* 30, 2 (2011), 14.

Alex Kendall, Vijay Badrinarayanan, , and Roberto Cipolla. 2015. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. *arXiv preprint arXiv:1511.02680* (2015).

Mateusz Kozinski, Raghudeep Gadde, Sergey Zagoruyko, Guillaume Obozinski, and Renaud Marlet. 2015. A MRF shape prior for facade parsing with occlusions. *IEEE CVPR* (2015), 2820–2828.

Florent Lafarge, Xavier Descombes, Josiane Zerubia, and Marc Pierrot-Deseilligny. 2010. Structural approach for building reconstruction from a single DSM. *IEEE TPAMI* 32, 1 (2010), 135–147.

Robert G Laycock and AM Day. 2003. Automatically generating large urban environments based on the footprint data of buildings. *ACM SMA* (2003), 346–351.

Minglei Li, Peter Wonka, and Liangliang Nan. 2016. Manhattan-world urban reconstruction from point clouds. *ECCV* (2016), 54–69.

Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. 2011a. Globfit: Consistently fitting primitives by discovering global relations. *ACM SIGGRAPH* 30, 4 (2011), 52.

Yangyan Li, Qian Zheng, Andrei Sharf, Daniel Cohen-Or, Baoquan Chen, and Niloy J Mitra. 2011b. 2D-3D fusion for layer decomposition of urban facades. *IEEE ICCV* (2011), 882–889.

Hui Lin, Jizhou Gao, Yu Zhou, Guiliang Lu, Mao Ye, Chenxi Zhang, Ligang Liu, and Ruigang Yang. 2013. Semantic decomposition and reconstruction of residential scenes from LiDAR data. *ACM SIGGRAPH* 32, 4 (2013), 66.

Anđelo Martinović, Markus Mathias, Julien Weissenberg, and Luc Van Gool. 2012. A three-layered approach to facade parsing. *ECCV* (2012), 416–429.

Bogdan C Matei, Harpreet S Sawhney, Supun Samarasekera, Janet Kim, and Rakesh Kumar. 2008. Building segmentation for densely built urban regions using aerial lidar data. *IEEE CVPR* (2008), 1–8.

Peter Miller et al. 2017. Buildings - OpenStreetMap Wiki. (2017). Retrieved August 8, 2017 from http://wiki.openstreetmap.org/wiki/Buildings

Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. 2015. RAPter: rebuilding man-made scenes with regular arrangements of planes. *ACM SIGGRAPH* 34, 4 (2015), 103–1.

Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. 2006. Procedural modeling of buildings. *ACM SIGGRAPH* 25, 3 (2006), 614–623.

Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. 2007. Image-based procedural modeling of facades. *ACM SIGGRAPH* 26, 3 (2007), 85.

Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, L v Gool, and Werner Purgathofer. 2013. A survey of urban reconstruction. *CGF* 32, 6 (2013), 146–177.

Liangliang Nan, Caigui Jiang, Bernard Ghanem, and Peter Wonka. 2015. Template assembly for detailed urban reconstruction. *CGF Eurographics* 34, 2 (2015), 217–228.

Liangliang Nan, Andrei Sharf, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2010. Smartboxes for interactive urban reconstruction. *ACM SIGGRAPH* 29, 4 (2010), 93.

Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Adrien Bousseau. 2016. Interactive sketching of urban procedural models. *ACM SIGGRAPH* 35, 4 (2016), 130.

Charalambos Poullis and Suya You. 2009. Automatic reconstruction of cities from remote sensor data. *IEEE CVPR* (2009), 2775–2782.

Hayko Riemenschneider, Ulrich Krispel, Wolfgang Thaller, Michael Donoser, Sven Havemann, Dieter Fellner, and Horst Bischof. 2012. Irregular lattices for complex shape grammar facade parsing. *IEEE CVPR* (2012), 1640–1647.

David Salinas, Florent Lafarge, and Pierre Alliez. 2015. Structure-Aware Mesh Decimation. *CGF* 34, 6 (2015), 211–227.

Chao-Hui Shen, Shi-Sheng Huang, Hongbo Fu, and Shi-Min Hu. 2011. Adaptive partitioning of urban facades. *ACM SIGGRAPH Asia* 30, 6 (2011), 184.

Noah Snavely, Steven M Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. *ACM SIGGRAPH* 25, 3 (2006), 835–846.

Olivier Teboul, Iasonas Kokkinos, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. 2013. Parsing facades with shape grammars and reinforcement learning. *IEEE TPAMI* 35, 7 (2013), 1744–1756.

Radim Tylecek. 2012. *The CMP facade database.* Technical Report. Tech. rep., CTU–CMP–2012–24, Czech Technical University.

Carlos A Vanegas, Daniel G Aliaga, and Bedřich Beneš. 2010. Building reconstruction using manhattan-world grammars. *IEEE CVPR* (2010), 358–365.

Yannick Verdie, Florent Lafarge, and Pierre Alliez. 2015. *LOD generation for urban scenes.* Technical Report 3. 30:1–30:14 pages. https://doi.org/10.1145/2732527

Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, and Long Quan. 2008. Image-based façade modeling. *ACM SIGGRAPH Asia* 27, 5 (2008), 161.

Chao Yang, Tian Han, Long Quan, and Chiew-Lan Tai. 2012. Parsing façade with rank-one approximation. *IEEE CVPR* (2012), 1720–1727.

Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J Mitra, Daniel Cohen-Or, and Baoquan Chen. 2010. Non-local scan consolidation for 3D urban scenes. *ACM SIGGRAPH* 29, 4 (2010), 94.

Qian-Yi Zhou and Ulrich Neumann. 2010. 2.5D dual contouring: a robust approach to creating building models from aerial lidar point clouds. *ECCV* (2010), 115–128.